Probabilistic Duration Automata for Analyzing Real-Time Systems

Louise E. Moser*

P. M. Melliar-Smith*

ABSTRACT [1] We present a novel methodology and tools for analyzing real-time systems that use probability density functions (pdfs) to represent the durations of operations within the system. We introduce the concept of a probabilistic duration automaton in which clocks are defined by pdfs rather than by explicit times. A state of a probabilistic duration automaton is a set of active clocks, and a transition is triggered by the expiration of one or more of these clocks. We present an algorithm for determining the probability that a clock in a state expires, the residual pdfs for the unexpired clocks, the probability of each transition, the probability of each state, and the duration of each state represented as a pdf. The algorithm also calculates the pdfs for durations of intervals between pairs of states within the automaton. These pdfs are used to determine whether a real-time system can meet its probabilistic timing constraints. An example application illustrates the use of this methodology in analyzing the real-time behavior of a four-phase handshaking protocol used in input/output systems.

1 Introduction

Traditionally, real-time systems have been defined by absolute requirements that certain events, typically input or output operations, must occur at or before precisely defined moments in time. To establish that such deadlines are met, existing methodologies and tools use worst-case upper bounds on the durations of operations within the system [8, 13, 14, 15].

Complex real-time systems, however, involve unreliable communication between distributed processors, asynchronous operations, unpredictable heuristic algorithms, and/or recovery from faults; furthermore, modern microprocessors exploit mechanisms such as caching and cycle stealing by

*Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

[1] This work was supported in part by ARPA Grant Nos. N00174-93-K-0097 and N00174-95-K-0083.

high-speed input/output devices. These characteristics introduce significant variations in the time required to produce the results. The worst-case upper bound for the duration of an operation may be substantially worse than the mean duration, but the worst-case upper bound may be realized only very infrequently.

A design based on the worst-case upper bound for the duration of every operation may be unnecessarily conservative, since it must allow for the possibility that each of these rare occurrences coincide. Thus, the performance calculated for the real-time system may be much worse than the performance that the system can actually provide. However, if the real-time system is designed using mean durations, occasionally the system may fail to meet a deadline. Such a failure to meet a deadline may be acceptable, provided that its probability is small enough and can be estimated reasonably accurately.

We describe a novel methodology and tools for analyzing real-time systems that represent the durations of operations by probability density functions (pdfs), rather than by upper and lower bounds. For some real-world applications in which real-time performance is critical, such as the Aegis air defense system, these pdfs are indeed known. The types of questions that our probabilistic analysis methodology can answer are

- What is the duration of the interval between two events, expressed as a probability density function?
- What is the probability of reaching a transient state, and what is the probability of being in a particular state in equilibrium?
- What is the probability that the duration of an operation exceeds the deadline for the operation?

As the basis for this methodology, we introduce the concept of a probabilistic duration automaton in which a clock is a probability density function. In each state of the automaton, several clocks may be active. A transition from one state to another state occurs when one or more active clocks expire; this may result in some of the expired clocks being reset.

We present an algorithm for evaluating such a probabilistic duration automaton. The algorithm determines the probabilities of the transitions out of a state, the rates of flow into a state, and the probability of being in a state. It also determines the probability that an active clock in a state expires, as well as the residual values of the clocks that do not expire, expressed as pdfs. The algorithm calculates the pdfs for the durations of the states and the pdfs for durations of intervals between pairs of states within the automaton. These pdfs are used to estimate the probability that a system can meet its real-time deadlines. Much of the interest of the analysis derives from the care required to ensure that two pdfs are convolved or combined only if they are independent.

As an application of this methodology, we consider the probabilistic timing constraints of a four-phase handshaking protocol used in input/output systems. This example demonstrates that manual analysis of these properties is difficult and that mechanical tools are essential.

2 The Clocks

In our methodology, a clock c is defined by a probability density function $\hat{c}$ that gives the probabilities for the possible durations of an interval between two events. The domain of the pdf, *i.e.,* the set of durations, is a finite set of positive rationals. Multiplication by the least common multiple of the denominators of these rationals yields a set of positive integers. Thus, without loss of generality, we assume that the durations are positive integers. We assume further that all of the pdfs have the same domain $\{1, 2, \ldots, m\}$ for some integer $m \geq 2$. The range of a pdf is a set of non-negative rationals between 0 and 1 such that the sum of these rationals is equal to 1. Thus, a clock c is defined by a pdf

$$\hat{c} : \{1, 2, \ldots, m\} \rightarrow [0, 1], \text{ where } \sum_{x=1}^{m} \hat{c}(x) = 1$$

For each duration x, $\hat{c}(x)$ is the probability that the interval from the time at which c was last set until it expires has duration exactly x.

Our methodology exploits the three operations of convolution, residue, and disjunction of pdfs. Convolution gives the pdf for the duration of an interval that is the sequential composition of two intervals, while residue gives the remainder of clock c_1, contingent on clock c_2 expiring before c_1, expressed as a pdf. Some intervals can be formed in several independent ways, each with a probability of occurrence and with its own pdf. Disjunction gives the pdf for the duration of an interval as the sum of these pdfs, weighted by their corresponding probabilities. These three operations are well-known and, thus, we do not define them here. When we employ these operations in our analysis, we must however ensure that the pdfs for the clocks being combined are indeed independent.

3 The Probabilistic Duration Automaton

A state of a probabilistic duration automaton is a set of active clocks. A transition from one state to another state in the automaton corresponds to the expiration of one or more active clocks. More specifically, we have the following definition.

A *probabilistic duration automaton* is defined by

- A finite set $S = \{s_i \mid 1 \leq i \leq N\}$ of states, a set $T \subseteq S \times S$ of transitions between states in S, and a set $C = \{c_i \mid 1 \leq i \leq M\}$ of clock names
- An initial state in S and a set $\hat{C} = \{\hat{c}_i \mid 1 \leq i \leq M\}$ of initial clock values (pdfs), one per clock name
- For each state $s_i \in S$, a set $A_{s_i} \subseteq C$ of clocks that are active in state s_i
- For each transition $(s_i, s_j) \in T$, a set $E_{s_i,s_j} \subseteq A_{s_i}$ of clocks whose expiration in state s_i triggers that transition.

All of the clocks in E_{s_i,s_j} expire at the same moment; none of the clocks in $R_{s_i,s_j} = A_{s_i} - E_{s_i,s_j}$ expires either before or at that moment. The clocks in R_{s_i,s_j} have residual values on the transition (s_i, s_j), and remain active in state s_j, *i.e.*, $R_{s_i,s_j} \subseteq A_{s_j}$. When a clock c is set or reset in any of the states, it is set to the initial clock value $\hat{c} \in \hat{C}$ for that clock. A clock must expire before it can be reset. The clocks that are set or reset on the transition (s_i, s_j) are the clocks in $A_{s_j} - R_{s_i,s_j}$.

Following the standard theory of Markov models, we consider two types of automata: equilibrium automata and transient automata. An *equilibrium automaton* is irreducible (every state can be reached from every other state) and aperiodic (there is no fixed period of returning to any state). The theory of Markov models tells us that, if any initial clock has two or more coprime durations with non-zero probabilities, then the automaton is aperiodic and the equilibrium probabilities of the states are time independent. A *transient automaton* has a single initial state, is reducible, and contains no irreducible subautomata. The probabilities of the states of a transient automaton are time dependent.

4 The Algorithm

A specification of a real-time system determines the set S of states and the set T of transitions of a probabilistic duration automaton, the set A_{s_i} of clocks that are active in state s_i, the set E_{s_i,s_j} of clocks whose expiration cause the transition (s_i, s_j), and the set $\hat{C}$ of pdfs that are the initial values of the clocks when they are set or reset.

Our algorithm determines the duration of each state or, equivalently, the time between the transitions into and out of that state, expressed as a pdf. It also determines the duration of the interval between each pair of states, from the time of the transitions into the first state to the time of the transitions into the second state, expressed as a pdf. To determine these pdfs, the algorithm calculates the probabilities of the transitions out of each state, the rates of flow into each state, the probability of being in a state, and related pdfs. The steps of the algorithm are presented below. Further explanation of the algorithm is given in Section 5.

4.1 Elaboration of the Automaton

The input to the algorithm is a probabilistic duration automaton obtained from the user's specification. If this automaton is reducible but contains an irreducible subautomaton, then the algorithm separates out that subautomaton and analyzes it separately. Such subautomata are replaced by absorbing states in the remaining automaton. Thus, the algorithm partitions the automaton into zero or more equilibrium automata and zero or more transient automata. If the analysis subsequently finds that some of the transition probabilities are zero, then it may be possible to partition the automaton further into simpler equilibrium automata and transient automata.

The algorithm must also eliminate correlations between pdfs before convolving or combining them. We define a state to be ***disjunctive*** if each of its in-edges has at most one unexpired clock and all of its in-edges have no unexpired clock or the same unexpired clock; otherwise, the state is ***non-disjunctive***. A state that is disjunctive has the desirable property that the residual pdfs on its in-edges can be combined. The algorithm splits non-disjunctive states into disjunctive states as outlined below.

1. Until each state with more than one in-edge is a disjunctive state, for each non-disjunctive state with more than one in-edge and for each such in-edge,
   a. Split that state by creating an additional state with
      i. The same set of active clocks as the given non-disjunctive state
      ii. A set of out-edges to the same destinations as the set of out-edges from the state being split, with corresponding in-edges having the same set of expiring clocks
   b. Change the destination of one of the in-edges to be the newly created state.
2. Partition the automaton into zero or more transient automata and zero or more equilibrium automata.
3. Set estimated rates of flow into each state and estimated residual pdfs for each unexpired clock.

Note that, after such elaboration, the automaton may still contain non-disjunctive states having one in-edge and two or more unexpired clocks on that in-edge.

4.2 Calculation for Individual States

For each state of a probabilistic duration automaton, the algorithm determines, the probability of taking a transition out of that state, the conditional pdfs for the duration of the state, the conditional pdfs for the unexpired clocks in the next state, and the rate of flow into the given state (Steps 1-3 below). From these quantities, the algorithm then derives the probability that a clock expires in the state, the pdf for the duration of the state, the pdf for each unexpired clock in the next state, and the probability of being in the given state (Steps 4-7 below).

For each automaton, until convergence is achieved,

1. For each disjunctive state s_i, calculate the residual pdf for the unexpired clock in state s_i from the residual pdfs for that clock on its in-edges, weighted by the rates of flow along those in-edges.

2. For each state s_i (disjunctive or not), trace backwards through the automaton, starting with s_i, until a disjunctive state is found. Denote this sequence of states by $\sigma_1, \ldots, \sigma_n$, where $\sigma_1 = s_h$ is the disjunctive state with the single unexpired clock c_h and $\sigma_n = s_i$. Note that this sequence is unique, that σ_k is non-disjunctive for $1 < k < n$ and that, if s_i is itself disjunctive, then the sequence contains the single state s_i. In the formulas below, we consider each set of possible durations $x_1, \ldots, x_n$ of the states $\sigma_1, \ldots, \sigma_n$ in this sequence.

   For each clock c that is active in some state of this sequence, let l_c be the lowest index in the sequence for which c is active and u_c the highest or upper index. For each clock c that is not active in any state of the sequence, set both l_c and u_c to 0. If a clock is reset at any point in the sequence and is active both before and after that point, regard that clock as two distinct clocks, one before the reset and one after. If c is the unexpired clock in the disjunctive state s_h, let $\hat{c}$ be the residual pdf for c in that state; otherwise, let $\hat{c}$ be the initial pdf corresponding to clock c when it is set or reset. If $l_c = u_c = 0$, extend the domain of $\hat{c}$ to include $x_0 = 0$ and set $\hat{c}(x_0) = 1$. If x is outside the domain of $\hat{c}$, set $\hat{c}(x) = 0$. As usual, interpret an empty sum to be 0 and an empty product to be 1. Set $\delta(p) = 1$ if the predicate p is true and $\delta(p) = 0$ otherwise.

3. For each edge (s_i, s_j) out of state s_i, calculate the following:

a. The probability of taking the edge (s_i, s_j) out of state s_i

$p_{s_i,s_j} =$

$$\frac{\displaystyle\sum_{x_1\ldots x_n}\left(\left(\prod_{c\in C-R_{s_i,s_j}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}{\displaystyle\sum_{\substack{x_1\ldots x_{n-1}\\ x_n=0}}\left(\left(\prod_{c\in C-A_{s_i}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in A_{s_i}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}$$

b. The probability of taking the edge (s_i, s_j) out of state s_i, contingent on the duration of the unexpired clock c_h at the start of state s_h being z_h,

$p_{s_i,s_j|z_h} =$

$$\frac{\displaystyle\sum_{x_1\ldots x_n}\left(\delta\Big(z_h=\sum_{l_{c_h}\leq k\leq u_{c_h}} x_k\Big)\times\left(\prod_{c\in B_R} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}{\displaystyle\sum_{\substack{x_1\ldots x_{n-1}\\ x_n=0}}\left(\delta\Big(z_h=\sum_{l_{c_h}\leq k\leq u_{c_h}} x_k\Big)\times\left(\prod_{c\in B_A} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in A_{s_i}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}$$

Here $B_R = (C - R_{s_i,s_j}) - \{c_h\}$ and $B_A = (C - A_{s_i}) - \{c_h\}$.

c. The pdf for the duration of state $\sigma_n = s_i$, contingent on the edge (s_i, s_j) being taken,

$\hat{d}_{s_i|s_i,s_j}(z) =$

$$\frac{\displaystyle\sum_{\substack{x_1\ldots x_{n-1}\\ x_n=z}}\left(\left(\prod_{c\in C-R_{s_i,s_j}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}{\displaystyle\sum_{x_1\ldots x_n}\left(\left(\prod_{c\in C-R_{s_i,s_j}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}$$

d. The pdf for the unexpired clock $a \in R_{s_i,s_j}$ in state s_j, contingent on the edge (s_i, s_j) being taken,

$\hat{a}_{s_j|s_i,s_j}(z) =$

$$\frac{\displaystyle\sum_{x_1\ldots x_n}\left(\hat{a}\Big(z+\sum_{l_a\leq k\leq u_a} x_k\Big)\times\left(\prod_{c\in C-R_{s_i,s_j}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}-\{a\}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}{\displaystyle\sum_{x_1\ldots x_n}\left(\left(\prod_{c\in C-R_{s_i,s_j}} \hat{c}\Big(\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\times\left(\prod_{c\in R_{s_i,s_j}}\sum_{x>0}\hat{c}\Big(x+\sum_{l_c\leq k\leq u_c} x_k\Big)\right)\right)}$$

e. From the probabilities p_{s_i,s_j} of the edges (s_i, s_j), calculate the rate f_{s_i} of flow into each state s_i by solving a system of linear flow-balance equations, in the usual manner.

4. From the probabilities p_{s_i,s_j} of the edges (s_i, s_j), calculate the probability that a clock c expires in state s_i by summing the probabilities of the edges on which c expires.

5. From the probabilities p_{s_i,s_j} of the edges (s_i, s_j) and the pdf $\hat{d}_{s_i|s_i,s_j}$ for the duration of state s_i, contingent on the edge (s_i, s_j) being taken, calculate the (unconditional) pdf $\hat{d}_{s_i}$ for the duration of state s_i as a weighted sum.

6. From the probabilities p_{s_i,s_j} of the edges (s_i, s_j) and the pdf $\hat{a}_{s_j|s_i,s_j}$ for the unexpired clock a in state s_j, contingent on the edge (s_i, s_j) being taken, calculate the (unconditional) pdf $\hat{a}_{s_j}$ for the unexpired clock a in state s_j as a weighted sum.

7. From the rate f_{s_i} of flow into each state s_i and the mean duration $\bar{d}_{s_i}$ of state s_i, calculate the probability of being in state s_i by multiplying f_{s_i} by $\bar{d}_{s_i}$ and then normalizing this product over all states.

Note that the equilibrium automata and the transient automata are handled differently in Step 3e, which involves the solution of the system of flow-balance equations, and in Step 7, which as given applies only to the equilibrium automata. For the transient automata, the probability of each non-absorbing state is 0 and the probability of each absorbing state is determined solely by the rate of flow into that state.

4.3 Calculation for Intervals between Pairs of States

The algorithm also determines the residual pdfs for the clocks and the conditional pdfs for the durations of paths and intervals between pairs of states (Steps 1-5 below). From these pdfs, the algorithm then derives the (unconditional) pdfs for the durations of the intervals between pairs of states (Step 6 below).

For each automaton, until convergence is achieved,

1. First consider a single sequence of states $\sigma_1, \ldots, \sigma_n, \sigma_{n+1}$, where $\sigma_1 = s_h$, $\sigma_n = s_i$ and $\sigma_{n+1} = s_j$. Here, the states s_h and s_j are disjunctive and the states σ_k, $1 < k < n+1$, are non-disjunctive. In Step 3 below, we calculate the pdf for the duration of this path. In Step 5, we calculate the pdf in the more general case, where there are multiple paths between s_h and s_j and the intermediate states of those paths may be either disjunctive or non-disjunctive. This can be generalized further to the case where s_h and s_j are not necessarily disjunctive.

Let the unexpired clocks in states s_h and s_j be c_h and c_j, respectively. We assume here that c_h and c_j are different clocks, *i.e.*, c_h is not the residual clock on the edge (s_i, s_j).

2. For each edge (s_i, s_j) out of state s_i, calculate the rate of flow on the edge (s_i, s_j), contingent on the residual duration of clock c_h at the start of state s_h being z_h and on the residual duration of clock c_j at the start of state s_j being z_j,

$$f_{s_i,s_j|z_h,z_j} = f_{s_h} \times \left(\prod_{1 \le k \le n} p_{\sigma_k,\sigma_{k+1}|z_h} \right) \times$$
$$\frac{\sum_{x_1 \ldots x_n} \left(\delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(z_j + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}{\sum_{x_1 \ldots x_n, x} \left(\delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(x + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}$$

3. For a single path between states s_h and s_j, calculate the pdf for the duration of that path from the start of s_h to the start of s_j, contingent on the residual duration of clock c_h at the start of s_h being z_h and on the residual duration of clock c_j at the start of s_j being z_j,

$$\hat{d}_{s_h,s_j|z_h,z_j}(y) =$$
$$\frac{\sum_{x_1 \ldots x_n} \left(\delta(y = \sum_{1 \le k \le n} x_k) \times \delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(z_j + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}{\sum_{x_1 \ldots x_n} \left(\delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(z_j + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}$$

4. Calculate the pdf for the unexpired clock c_j in state s_j, contingent on the edge (s_i, s_j) being taken and on the residual duration of clock c_h at the start of state s_h being z_h,

$$(\hat{c}_j)_{s_j|s_i,s_j,z_h}(z) =$$
$$\frac{\sum_{x_1 \ldots x_n} \left(\delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(z + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}{\sum_{x_1 \ldots x_n, x} \left(\delta(z_h = \sum_{l_{c_h} \le k \le u_{c_h}} x_k) \times \hat{c}_j(x + \sum_{l_{c_j} \le k \le u_{c_j}} x_k) \times \prod_{c \in C - \{c_h, c_j\}} \hat{c}(\sum_{l_c \le k \le u_c} x_k) \right)}$$

5. For the interval (aggregate of paths) between states s_h and s_j, calculate the pdf for the duration of the interval from the start of s_h to the start of s_j, *contingent on the residual duration of clock c_h at the start of s_h being z_h and on the residual duration of clock c_j at the start of s_j being z_j*,

$$\hat{d}_{s_h,s_j|z_h,z_j}(y) = \frac{\sum_{s_l,y_1,y_2} \delta(y = y_1 + y_2) \times f_{s_l,s_j|z_h,z_j} \times \hat{d}_{s_h,s_l|z_h,z_l}(y_1) \times \hat{d}_{s_l,s_j|z_l,z_j}(y_2)}{\sum_{s_l} f_{s_l,s_j|z_h,z_j}}$$

Here one state s_l is chosen for each path from s_h to s_j.

6. For the interval (aggregate of paths) between states s_h and s_j, calculate the pdf for the duration of the interval from the start of s_h to the start of s_j as the weighted sum

$$\hat{d}_{s_h,s_j}(y) = \sum_{z_h,z_j} \hat{d}_{s_h,s_j|z_h,z_j}(y) \times \hat{c}_h(z_h) \times (\hat{c}_j)_{s_j|s_i,s_j,z_h}(z_j)$$

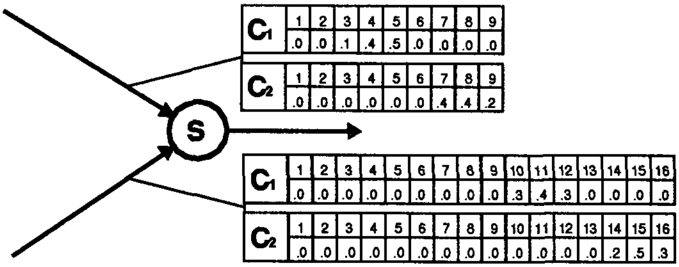
5 Discussion

We now provide further explanation of the algorithm, with examples to illustrate the problems that can arise if correlated pdfs are not handled properly.

5.1 Eliminating Correlations

The main intricacies of the algorithm arise from the need to eliminate correlations between pdfs that are being combined. For example, consider a state s with two in-edges, where two independent clocks c_1 and c_2 were active in both of the prior states and did not expire, as shown in Fig. 1. Along one edge the residual pdf for c_1 is non-zero for 3 to 5 and the residual pdf for c_2 is non-zero for 7 to 9, while along the other edge the residual pdf for c_1 is non-zero for 10 to 12 and the residual pdf for c_2 is non-zero for 14 to 16. Naively combining the two residual pdfs for c_1 from the two in-edges into a single pdf, and similarly for c_2, would lead to an erroneous conclusion such as that c_2 expires before c_1 or that when c_1 expires the residual pdf for c_2 is non-zero for 2 to 13. This example demonstrates the need to split non-disjunctive states with two or more in-edges, each of which has two or more unexpired clocks, because there is a dependence between the unexpired clocks on those in-edges.

Next consider the example shown in Fig. 2. Here, clocks c_1 and c_2 are active in the initial state s_1. If c_2 expires before c_1, the transition is taken to state s_2 whereas, if c_1 expires before c_2, the transition is taken to state s_3.



C1	1	2	3	4	5	6	7	8	9
	.0	.0	.1	.4	.5	.0	.0	.0	.0

C2	1	2	3	4	5	6	7	8	9
	.0	.0	.0	.0	.0	.0	.4	.4	.2

C1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	.0	.0	.0	.0	.0	.0	.0	.0	.0	.3	.4	.3	.0	.0	.0	.0

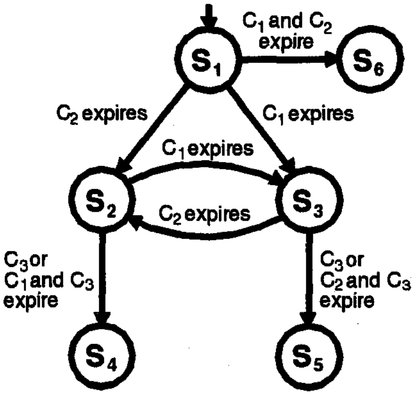
C2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0	.2	.5	.3

FIGURE 1. An example showing the need to split a non-disjunctive state. In this example, clock c_2 cannot expire before clock c_1 in state s. To ensure this, the state is split, avoiding a disjunction of correlated pdfs.

If c_1 and c_2 expire simultaneously, the transition is taken to state s_6. Clock c_3 becomes active in states s_2 and s_3. If c_1 expires before c_3 in s_2, the transition is taken to s_3 and c_2 becomes active. If c_2 expires before c_3 in s_3, the transition is taken back to s_2 and c_1 becomes active again. If c_3 expires before or simultaneously with c_1 in s_2, the transition is taken to s_4. Similarly, if c_3 expires before or simultaneously with c_2 in s_3, the transition is taken to s_5. Fig. 3 shows the automaton of Fig. 2 after state splitting. It shows the probability of each transition, the probabilities of reaching states s_4, s_5 and s_6, and the pdfs for the durations until those states are reached from s_1.

Now consider the example shown in Fig. 4. Here we have a state s_1 in which three clocks become active: c_1 is non-zero for 2 to 4, c_2 is non-zero for 6 to 10, and c_3 is also non-zero for 6 to 10. When clock c_1 expires, the transition to state s_2 is taken and, when clock c_2 expires, the transition to state s_3 is taken. The residual pdf for c_2 in s_2 is non-zero for 2 to 8, as is the residual pdf in s_2 for c_3. It is incorrect to combine these pdfs and to conclude that the residual pdf for c_3 in s_3 is non-zero for 1 to 6, because they are both obtained from the pdf for c_1 in s_1. This example demonstrates the need to trace backwards through the automaton until a disjunctive state is found, because states with more than one unexpired clock on their in-edges may introduce dependencies between those clocks.

Finally, consider the example in Fig. 5, which illustrates the problem of determining the pdf for the duration of an interval by convolving together the pdfs for the durations of its component states. Here, since c_1 necessarily expires before c_2, the pdf for the duration of state s_1 is non-zero for 1 to 5. Consequently, the pdf for the duration of state s_2, being the residual pdf of c_2 after c_1 has expired, is non-zero for 1 to 9. Simple convolution of the pdfs for the durations of s_1 and s_2 would lead to the incorrect conclusion that the pdf for the duration of the interval consisting of those two states is non-zero for 2 to 14 whereas, in fact, the pdf for that duration is non-zero for 6 to 10.



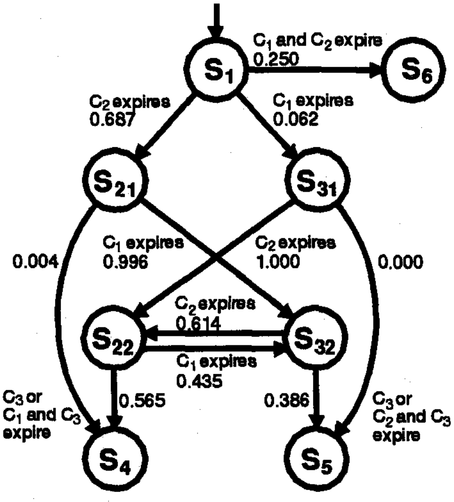
Initial Clock Values

	1	2	3	4	5
C_1	.000	.000	.250	.500	.250

	1	2	3	4
C_2	.000	.250	.500	.250

	1	2	3	4	5	6	7	8	9	10	11	12
C_3	.000	.000	.100	.100	.100	.100	.100	.100	.100	.100	.100	.100

FIGURE 2. A simple probabilistic duration automaton involving six states and three clocks. The initial values of the clocks are also shown.



		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S_1 to S_4	.377	.000	.000	.000	.000	.017	.021	.054	.120	.169	.172	.141	.093	.060	.071	.071	.011

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S_1 to S_5	.373	.000	.000	.000	.000	.050	.163	.146	.080	.030	.027	.058	.108	.140	.129	.062	.007

		1	2	3	4
S_1 to S_6	.250	.000	.000	.500	.500

FIGURE 3. The automaton of Fig. 2 after state splitting. For each of the transitions, the probability of that transition is shown. For each of the states s_4, s_5 and s_6, the probability of reaching that state and the pdf for the duration until that state is reached from s_1 are also shown.

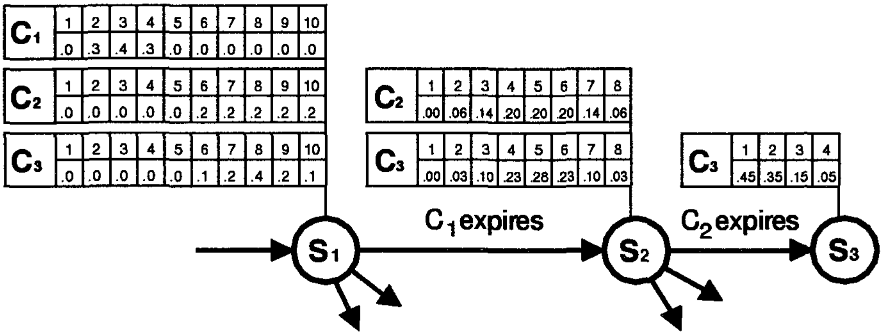


FIGURE 4. Clock c_1 expires in state s_1. Because the residual pdfs for clocks c_2 and c_3 in state s_2 are correlated, it is incorrect to derive the residual pdf for clock c_3 in state s_3 from the residual pdfs for c_2 and c_3 in state s_2.

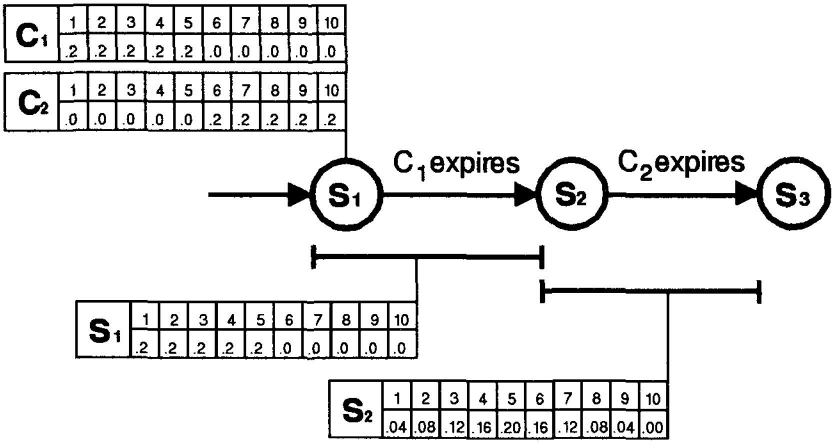


FIGURE 5. The pdfs for the durations of states s_1 and s_2 are correlated. It is incorrect to derive the pdf for the duration of the interval consisting of these two states by convolving the pdfs for their individual durations.

5.2 Disjunctive States

For a disjunctive state, such correlation problems do not arise. We can combine the residual pdfs for the unexpired clock on each of its in-edges into a single pdf, weighting the contribution of the residue on each in-edge by the relative flow of that in-edge.

We are investigating an extension of the algorithm that would allow a disjunctive state to have multiple in-edges and multiple unexpired clocks on those in-edges, provided that all of its in-edges have the same set of unexpired clocks and, for each such clock, the residual pdfs on the in-edges for that clock are equal, with the required or available degree of precision.

Disjunctive states have the advantage over pure state splitting that they substantially reduce the size of the automaton. Unlike the state splitting used in real-time temporal logic decision procedures [14], we can combine states when the pdf for an unexpired clock is a disjunctive composition, whereas such a decision procedure must continue to split states.

5.3 Calculating the Probabilities

As the above examples indicate, we must be careful in calculating the probability p_{s_i,s_j} that the edge (s_i, s_j) is taken when the clocks in E_{s_i,s_j} expire. The analysis must commence with the values determined for those clocks in a disjunctive state or assigned when the clocks were set or reset in a subsequent state, and then consider the effect of the durations of the states since the clocks were determined or set. Fortunately, for each non-disjunctive state, we need only consider a single sequence of prior states since, after state splitting, states with two or more in-edges are disjunctive.

In the numerator for p_{s_i,s_j}, we sum over all of the possible durations $x_1, \ldots, x_n$ of the states in the sequence of prior states, as illustrated by the example in Fig. 6 where $(s_i, s_j) = (s_3, s_4)$, s_1 is the disjunctive state, and c_1 is the unexpired clock on entering that state. Within this summation are two products. The first represents the probability that the expiring clocks (c_1, c_2 and c_3 in Fig. 6) expire at the ends of the states in which they are presumed to expire. Note that a clock c that is set in state σ_{l_c} of the sequence and that expires in state σ_{u_c} has been running for a duration that is the sum of the durations $x_{l_c}, \ldots, x_{u_c}$, as represented by the inner summation. The probability that clock c expires at exactly this moment is determined by the pdf determined for c in the disjunctive state or assigned to c when it was set. The second product provides the probability that the non-expiring clocks (c_4 and c_5 in Fig. 6) do not expire, and is obtained by summing over all larger durations. The denominator for p_{s_i,s_j} represents the probability of reaching state s_i (s_3 in Fig. 6).

The probability density function $\hat{d}_{s_i|s_i,s_j}$ for the duration of state s_i, contingent on the edge (s_i, s_j) being taken, is a variation of the formula for the probability p_{s_i,s_j} of that edge. To determine this pdf, we consider sepa-

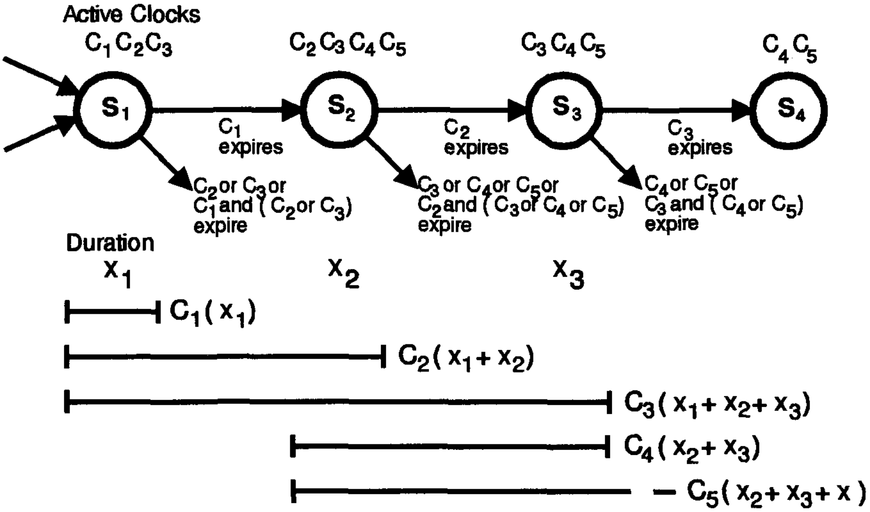


FIGURE 6. An example of a sequence of states preceding the transition (s_3, s_4). State s_1 is a disjunctive state. The figure shows the clocks that are active in each state, the clocks that expire on each transition, and the duration of each state.

rately each possible duration $z = x_n$ for state $s_i = \sigma_n$. Consequently, x_n is excluded from the outer summation in this formula and instead appears as the duration z on the left of the equality. Similarly, the residual pdf $\hat{a}_{s_j|s_i,s_j}$ for clock a in state s_j, contingent on the edge (s_i, s_j) being taken, is obtained by excluding clock a from the last product and instead considering explicitly, as the first factor in the outer product, the probability that clock a survives for an additional duration z.

To calculate the pdfs for the durations of longer intervals, we combine together the durations of shorter intervals, typically the pdfs for the durations from one disjunctive state to the next. Unfortunately, as illustrated in Fig. 5, simple convolution of these pdfs may be incorrect because they may be dependent. Consider the composition of the interval from s_h to s_l and the interval from s_l to s_j, where there is only one unexpired clock c_l on the transition into s_l. We can convolve the pdf for the duration of the interval from s_h to s_l, contingent on c_l having residual duration z_l at the start of s_l, and the pdf for the duration of the interval from s_l to s_j, contingent on c_l having the same residual duration z_l at the start of s_l, since their dependency is fully constrained by that contingency.

Thus, for two disjunctive states s_h and s_j with unexpired clocks c_h and c_j, respectively, we define $\hat{d}_{s_h,s_j|z_h,z_j}$ to be the pdf for the duration of the interval from s_h to s_j, contingent on c_h having residual duration z_h at the start of s_h and on c_j having residual duration z_j at the start of s_j. We make this pdf contingent on the residual durations of the unexpired clocks in both the initial and final states so that we can convolve the pdf for the duration of this interval with the pdf for the duration of either the preceding interval or of the following interval without risk of incorrectness. If more than one path exists from s_h to s_j, the pdfs for the durations of those paths are weighted by the rates of flow along those paths. We then calculate the (uncontingent) pdf $\hat{d}_{s_h,s_j}$ for the duration of the interval from s_h to s_j from the contingent pdfs for that duration and the residual pdfs for the unexpired clocks in the states s_h and s_j.

5.4 Termination

The termination of the state splitting procedure may depend on an approximation, justified by the limited precision of the representation of real numbers in the computer and the uncertainty of the exact values of pdfs in the real world. We are investigating the possibility of not splitting a state if, for each unexpired clock, the differences in the residual pdfs for that clock on the in-edges to that state are sufficiently small. If the analysis shows that we have erred in our estimate of the states that do not need to be split and that for some unexpired clock the pdfs on the in-edges to some state differ significantly, the automaton is enlarged and the analysis is repeated.

6 An Example Application

As an example application, we consider the probabilistic timing constraints of the classical four-phase handshaking protocol used in input/output systems. In its simplest form, the protocol involves two agents, a requester (the processor) and a responder (the device).

The requester sets

- *addr*: a predicate representing the presence of address information on the address bus
- *req*: a boolean control signal indicating to the responder that the requester has placed address information on the address bus.

The responder sets

- *data*: a predicate representing the presence of data on the data bus
- *resp*: a boolean control signal indicating that the responder has received the requester's address information and that the responder has placed information on the data bus for the requester.

Initially, all four of these signals are false.

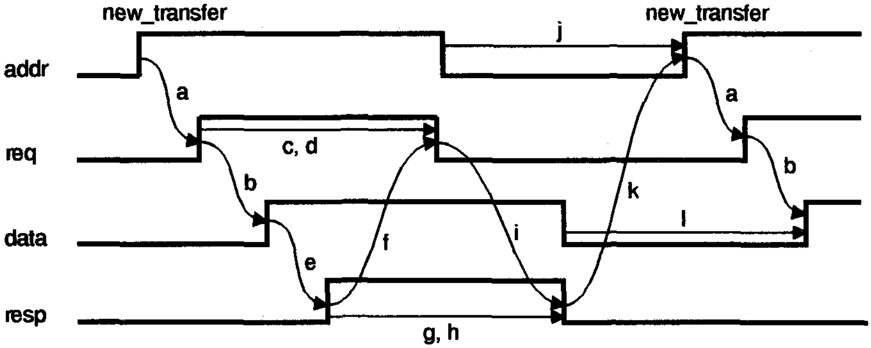


FIGURE 7. The timing diagram for the four-phase handshaking protocol, labeled with clocks. The clocks are pdfs that represent the durations of the indicated intervals.

Duration Clocks	1	2	3	4	5	6	7	8	9	10
a,e	.5	.5	.0	.0	.0	.0	.0	.0	.0	.0
c,d,g,h,j,l	.0	.0	.0	.0	.0	.0	.0	.0	.5	.5
b,f,i,k	.0	.2	.4	.2	.1	.05	.02	.01	.01	.01

TABLE .1. The initial values of the clocks, given as pdfs.

The protocol operates in four phases:

1. The requester places information on the address bus and, after a short delay, sets *req* to true.

2. The responder detects that *req* has become true and reads the address information on the address bus. It then places the requested information on the data bus and, after a short delay, sets *resp* to true.

3. The requester detects that *resp* has become true and reads the information on the data bus. At this point, it knows that the responder has detected that *req* has become true and has read the information on the address bus. The requester then sets *addr* and *req* to false.

4. The responder detects that *req* has become false and knows that the requester has read the information on the data bus. The responder then sets *data* and *resp* to false.

Once the requester has detected that *resp* has become false, the requester can restart the cycle by placing further information on the address bus.

The timing diagram is shown in Fig. 7. Each thick time line corresponds to one of the four predicates. The thinner lines with arrowheads are labeled

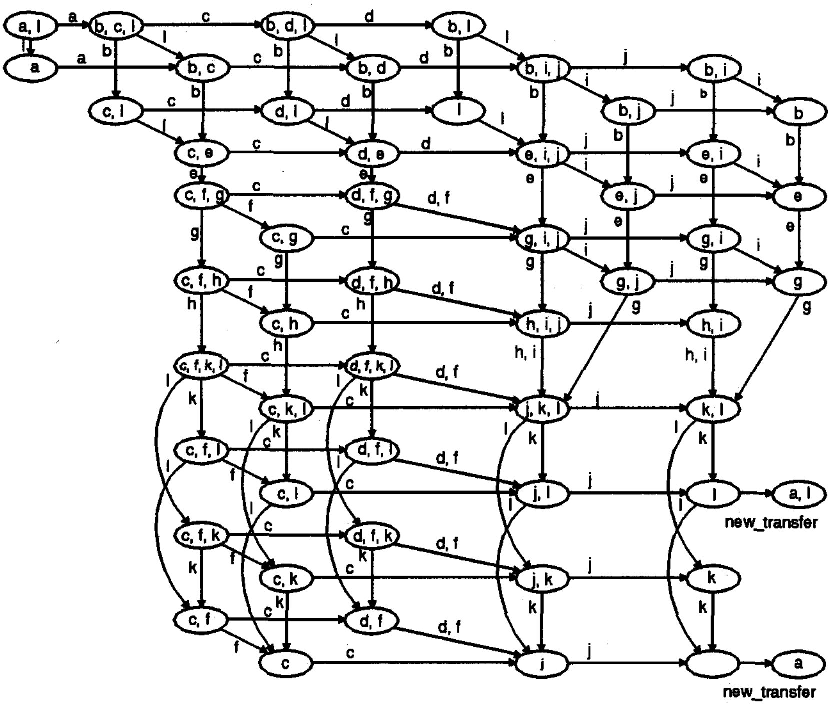


FIGURE 8. The probabilistic duration automaton for the four-phase handshaking protocol. The states are labeled with the active clocks, and the edges with the expiring clocks.

with clocks that define the durations between the indicated transitions. The initial values of the clocks are given in Table 1 as pdfs.

Clock *c* specifies a lower bound on the duration of the indicated interval; clock *d*, which is started when clock *c* expires, imposes an upper bound. When clocks *c* and *f* expire, *d* becomes inactive and, when clocks *c* and *d* expire, *f* becomes inactive. Similarly, when clocks *g* and *h* expire, *i* becomes inactive and, when clocks *g* and *i* expire, *h* becomes inactive.

Clocks *j* and *l* specify the lower bounds on the durations for which the signals *addr* and *data* are false, respectively, and impose no upper bounds on these durations. The signal *addr* becomes true when both clocks *j* and *k* expire and, in addition, the external predicate *new_transfer* is true. Similarly, the signal *data* becomes true when both clocks *b* and *l* expire.

The probabilistic duration automaton for the four-phase handshaking protocol is shown in Fig. 8. The states of the automaton are labeled with the active clocks, and the edges with the expiring clocks. The two states at the bottom right of the diagram are the same as the two states at the top left and, thus, the automaton is an equilibrium automaton.

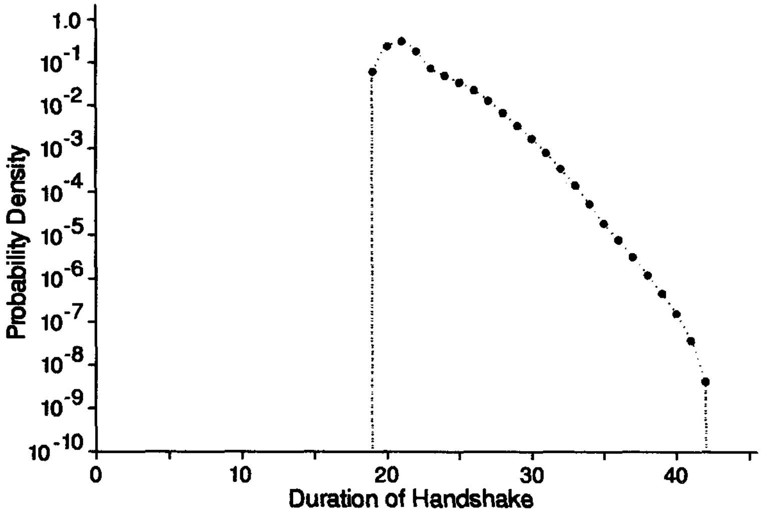


FIGURE 9. The calculated pdf for the duration of the handshake from all four signals being false until all four signals are again false. Note the logarithmic scale on the vertical axis.

The automaton shown in Fig. 8 is a simplification of the full automaton in that the edges that represent simultaneous expiration of clocks are not shown and state splitting has not been performed. After state splitting, the automaton had grown to 262 states and 1162 edges. The calculated pdf for the duration of the handshake is shown in Fig. 9.

Other tools are able to predict the maximum duration of 42, but we are aware of no other tool that can predict the probability of a particular duration for the handshake, a probability that might be of great interest to the designers of real-time systems. The computation (highly unoptimized) required 1 minute 25 seconds on a Sun SPARCstation 20. Only 31 of the states and 81 of the edges had non-zero rates of flow.

7 Related Work

Much work has been done on establishing that real-time systems can meet their deadlines using worst-case analysis with upper and lower bounds, for example [8, 15]. This work diverges from that view, and is more closely related to the work of [1, 3], which develops verification techniques for probabilistic real-time systems. It derives from our work on real-time interval temporal logic [13, 14], and we extend the Büchi automata [2, 16] upon which a decision procedure for such a temporal logic is based. Unlike prior researchers, we introduce the notion of a clock as a pdf, and use pdfs to represent the durations of states, as well as the durations of intervals between pairs of states within the automaton.

Various researchers have developed methodologies for reasoning about probabilistic programs and randomized algorithms. Feldman [4] and Kozen [10] have defined propositional probabilistic dynamic logics, where events occur according to a probability distribution. Courcoubetis and Yannakakis [3] have investigated probabilistic linear-time temporal logics, where probabilistic programs are modeled using Markov chains, and have devised a model-checking algorithm to determine whether a program satisfies its specification in the logic. Hansson and Jonsson [5] have extended the CTL branching-time logic with time and probabilities, and have also provided a model-checking algorithm to determine whether a given Markov chain satisfies a formula of the logic.

Liu, Ravn, Sorensen and Zhou [11] have developed a probabilistic duration calculus for reasoning about, and calculating whether, a given requirement of a real-time system holds with a sufficiently high probability for given failure probabilities of the system. Their calculus is a real-time interval temporal logic in which the user specifies requirements of the system and defines satisfaction probabilities for formulas of the logic. The system model is a finite automaton with fixed transition probabilities; discrete Markov processes are the basis of the calculus.

Jonsson and Larsen [9] have investigated probabilistic transition systems for describing and analyzing reliability aspects of concurrent distributed systems. In [7] Huang, Lee and Hsu have presented a similar extended state transition model, Timed Communicating State Machines, for protocol verification. Ho-Stuart, Zedan and Fang [6] have also described a tool, called the CoS-Workbench, for analyzing and manipulating formal specifications of real-time processes, interpreted as labeled transition systems.

In [12] Lynch, Saias and Segala defined probabilistic timed automata as an extension to I/O automata, which address such questions as whether an algorithm will terminate within time t with probability p. Wu, Smolka and Stark [18] have also introduced probabilistic I/O automata for specifying and reasoning about asynchronous probabilistic systems. However, none of the above approaches uses pdfs to represent clocks, as we do here.

In [1] Alur, Courcoubetis and Dill presented a model-checking algorithm for a probabilistic real-time system modeled as a generalized semi-Markov process, where a specification of the system is given as a deterministic timed automaton. They associate a pdf with each delay and, thus, their approach is similar to ours; however, they assume the existence of a generalized semi-Markov process, whereas we construct it from a given set of pdfs. The completed probabilistic duration automaton constructed by our algorithm can be used as input to their model-checking algorithm. Much closer to our work is the work of Whitt [17], who has investigated the convergence of the construction of generalized semi-Markov processes.

8 Conclusion

We have described a novel methodology and tools for analyzing probabilistic timing properties of real-time systems. In our methodology, clocks are defined as probability density functions, and probabilistic duration automata are defined in terms of the pdfs to which the clocks are set, the clocks that are active in each state, and the clocks that expire on each transition.

The algorithm, which we have implemented, takes as input a probabilistic duration automaton. The probabilistic duration automaton may itself be the user's specification of the real-time system or may be obtained from a temporal logic specification. The algorithm determines the probabilities that clocks in a state expire, the residual pdfs for the other clocks, the probabilities of the transitions, the probabilities of the states, and the pdfs for the durations of the states. The algorithm also determines the pdfs for durations of intervals between pairs of states within the automaton. These pdfs are used to determine whether a real-time system can meet its probabilistic timing constraints.

We plan to integrate this algorithm with a decision procedure for satisfiability checking of interval temporal logic formulas in which the durations of intervals are expressed as pdfs. The decision procedure for this interval temporal logic will construct the automaton from the interval temporal logic formula in the usual manner. Our algorithm will complete this automaton by finding the residual pdfs for the clocks, and the related probabilities and pdfs for the durations. We also plan to extend this methodology to handle continuous pdfs, but different techniques will be required.

9 REFERENCES

[1] R. Alur, C. Courcoubetis and D. Dill, "Verifying automata specifications of probabilistic real-time systems," *Proceedings of the REX Workshop on Real-Time: Theory in Practice*, Mook, The Netherlands (June 1991), Lecture Notes in Computer Science 600, Springer-Verlag, pp. 28-44.

[2] R. Alur and D. Dill, "Automata for modelling real-time systems," *Proceedings of the 17th International Colloquium on Automata, Languages, and Programming*, Warwick, England (July 1990), Lecture Notes in Computer Science 443, Springer-Verlag, pp. 322-335.

[3] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *Journal of the Association for Computing Machinery* 42, 4 (July 1995), pp. 857-907.

[4] Y. A. Feldman, "A decidable propositional dynamic logic with explicit probabilities," *Information and Control* 63 (1984), pp. 11-38.

[5] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal Aspects of Computing* 6, 5 (1994), pp. 512-535.

[6] C. Ho-Stuart, H. Zedan and M. Fang, "Automated support for the formal specification and design of real-time systems," *Proceedings of the Nineteenth EUROMICRO Symposium on Microprocessing and Microprogramming*, Barcelona, Spain (September 1993), pp. 79-86.

[7] C. M. Huang, S. W. Lee and J. M. Hsu, "Probabilistic timed protocol verification for the extended state transition model," *Proceedings of the 1994 International Conference on Parallel and Distributed Systems*, Hsinchu, Taiwan (December 1994), pp. 432-437.

[8] S. Jahanian and A. K. Mok, "Modechart: A specification language for real-time systems," *IEEE Transactions on Software Engineering* 20, 11 (November 1994), pp. 933-947.

[9] B. Jonsson and K. G. Larsen, "Specification and refinement of probabilistic processes," *Proceedings of the 6th IEEE Annual Symposium on Logic in Computer Science*, Amsterdam, The Netherlands (July 1991), pp. 266-277.

[10] D. Kozen, "Probabilistic PDL," *Journal of Computer and System Sciences* 30 (1985), pp. 162-178.

[11] Z. Liu, A. P. Ravn, E. V. Sorensen and C. Zhou, "A probabilistic duration calculus," *Proceedings of the Second International Workshop on Responsive Computer Systems*, Saitama, Japan (October 1992), pp. 14-27.

[12] N. Lynch, I. Saias and R. Segala, "Proving time bounds for randomized distributed algorithms," *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, Los Angeles, CA (August 1994), pp. 314-323.

[13] L. E. Moser, Y. S. Ramakrishna, G. Kutty, P. M. Melliar-Smith and L. K. Dillon, "A graphical environment for design of concurrent real-time systems," Technical Report 95-18, Department of Electrical and Computer Engineering, University of California, Santa Barbara.

[14] Y. S. Ramakrishna, L. K. Dillon, L. E. Moser, P. M. Melliar-Smith and G. Kutty, "A real-time interval logic and its decision procedure," *Proceedings of the Thirteenth Conference on Foundations of Software Technology and Theoretical Computer Science*, Bombay, India (December 1993), Lecture Notes in Computer Science 761, Springer-Verlag, pp. 173-192.

[15] A. C. Shaw, "Communicating real-time state machines," *IEEE Transactions on Software Engineering* 18, 9 (September 1992), pp. 805-816.

[16] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification," *Proceedings of the Symposium on Logic in Computer Science*," Cambridge, England (June 1986), pp. 322-331.

[17] W. Whitt, "Continuity of generalized semi-Markov processes," *Mathematics of Operations Research* 5, 4 (November 1980), pp. 494-501.

[18] S. H. Wu, S. A. Smolka and E. W. Stark, "Composition and behaviors of probabilistic I/O automata," *Proceedings of the 5th International Conference on Concurrency Theory*, Uppsala, Sweden (August 1994), pp. 513-528.